University of Victoria
CSC 360: Operating Systems
Dr. Jianping Pan

ASSIGNMENT P2A
MULTI-THREADED SCHEDULER DESIGN DOCUMENT

Due: February 21st, 2018

William H. Grosset
V00820930

1.
N + 1 threads will be used, where N represents how many trains will be in the input file and the + 1 being the main thread. Each train thread will be in charge of its own loading and crossing time and the main thread is in charge of dispatching trains (at most 1) across the track.

2.
Trains load concurrently via threads, but the main thread acts as the controller thread. The main thread decides when all trains will be initially loaded and when a train will cross the track.

3.
- Load mutex (guards access to begin loading)
- Ready mutex (guards access to be next released train)
- Track mutex (guards access to the track)
- West station mutex (guards access to the west station priority queue)
- East station mutex (guards access to the west station priority queue)

4.
The main thread is either dispatching trains or waiting for a signal from a station (idle).

5.
Priority (FIFO) queue for **each** station to represent the trains that are ready to cross.

6.
A mutex to guard multiple threads from being able to enqueue/dequeue from a station priority queue.

7.
N + 2 condition variables, where N is the number of trains/threads (each have `canCross`), and `canLoad` and `isReady` are the 2 global condition variables.

Convar: `canLoad`

a) The condition represents if a train can begin loading.
b) Load mutex — gateway to begin loading
c) After `pthread_cond_wait()`: Each thread will begin to `usleep` for their appropriate loading time and then are added to the station priority queue.

Convar: `isReady`

d) The condition represents if a train is ready in the station queue.
e) Ready mutex — dispatcher waits to be signaled by a ready train

f) After `pthread_cond_wait()`: The main thread will `peek()` on both queues and decide which train to release (based on the simulation rules) to cross the track.

Convar: `canCross`

g) The condition represents if a train can cross.
h) Track mutex — only one thread can access the track
i) After `pthread_cond_wait()`: The targeted thread will begin to `usleep` for their appropriate crossing time and then signals the main thread once finished.

8.

Main Thread = MT; Train Thread = TT

- MT: Read input file, create threads for each train
- TT: Lock load mutex, wait for signal to load train
- MT: Lock load mutex, broadcast to all trains to load, release mutex
- MT: Lock ready mutex, wait to be signaled by a ready train
- TT: Release load mutex, load (`usleep`)
- TT: Lock station mutex, enqueue to station priority queue, release mutex
- TT: Lock ready mutex, signal to main thread that this train is ready, release mutex
- MT: Release ready mutex, check both station priority queues and choose appropriate train (based of simulation rules)
- MT: Lock station mutex, dequeue from station priority queue, release mutex
- MT: Main thread signals train thread
- MT: Wait for train to signal after finished crossing
- TT: Lock track mutex
- TT: Cross (`usleep`)
- TT: Signal finish and release track mutex and
- MT: Destroy thread and exit if all trains have crossed